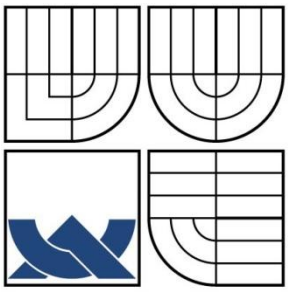
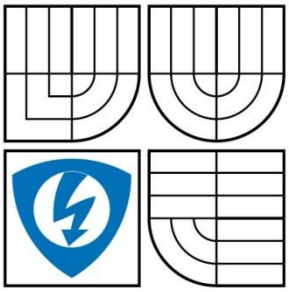


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKACNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

KOMPRESNÍ TECHNIKY STATICKÉHO OBRAZU

STATIC IMAGE COMPRESSION TECHNIQUES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MATĚJ JIROUNEK**
AUTHOR

VEDOUCÍ PRÁCE **ING. ONDŘEJ KRAJSA**
SUPERVISOR

BRNO 2009

ANOTACE

Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

KOMPRESNÍ TECHNIKY STATICKÉHO OBRAZU

Bakalářská práce

Zaměření studia: Teleinformatika

Autor: Matěj Jirounek

Vedoucí práce: Ing. Ondřej Krajsa

Abstrakt:

Bakalářská práce pojednává o dnes používaných kompresních technikách statického obrazu, o jejich základních principech, výhodách a nevýhodách v oblasti použití a jejich vzájemném porovnání. Práce je rozdělena do sedmi kapitol, z nichž druhá pojednává rozdělení komprese dat, třetí o bezztrátové kompresi a jejich hlavními představiteli a čtvrtá naopak o ztrátové kompresi obrazu s jejími výhodami a nedostatky a také o barevných modelech. V páté kapitole jsou shrnuty použitá kritéria pro hodnocení obrazu a v šesté kapitole je ukázána implementace programu v prostředí MATLAB.

Klíčová slova: Komprese, JPEG, JPEG2000, DFT, DCT, Vektorová kvantizace, barevné modely, MATLAB

ABSTRAKT

Brno University of Technology

Faculty of Electrical Engineering and Communication

Department of telecommunication

STATIC IMAGE COMPRESSION TECHNIQUES

Bachelor's thesis

Specialisation of study: Teleinformatics

Student: Matěj Jirounek

Supervisor: Ing. Ondřej Krajsa

Abstract:

Bachelor's thesis dissert on nowadays used compression technologies static image, about their keystones, advantages and disadvantages in field of application and their reciprocal comparison. Work is divided to the seven capitols, second of which handles indicative allocation data compression, third about lossless compression and their main representative and fourth chapter is about less compression picture with her advantages and disadvantages as well as about colour models. In fifth chapter there are summarized used criteria for classification picture and in sixth chapter is shownd implementation programme in environment MATLAB.

Keywords: Compression, JPEG, JPEG2000, DFT, DCT, Vector quantization, colour models, MATLAB

JIROUNEK, M. *Kompresní techniky statického obrazu.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 54 s. Vedoucí bakalářské práce Ing. Ondřej Krajsa.

Prohlášení

Prohlašuji, že svoji bakalářskou práci na téma *„Kompresní techniky statického obrazu“* jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ..... …… podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Ondřeji Krajsovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne ....

(podpis autora)

OBSAH

Seznam obrázků

1 Úvod

Cílem této práce: „Kompresní techniky statického obrazu“ je se seznámit s metodami, jakými můžeme obrazová data komprimovat, shrnou jejich klady a zápory a poukázat na oblast využití. V této práci se zaměřím především na standard JPEG a standard JPEG 2000.

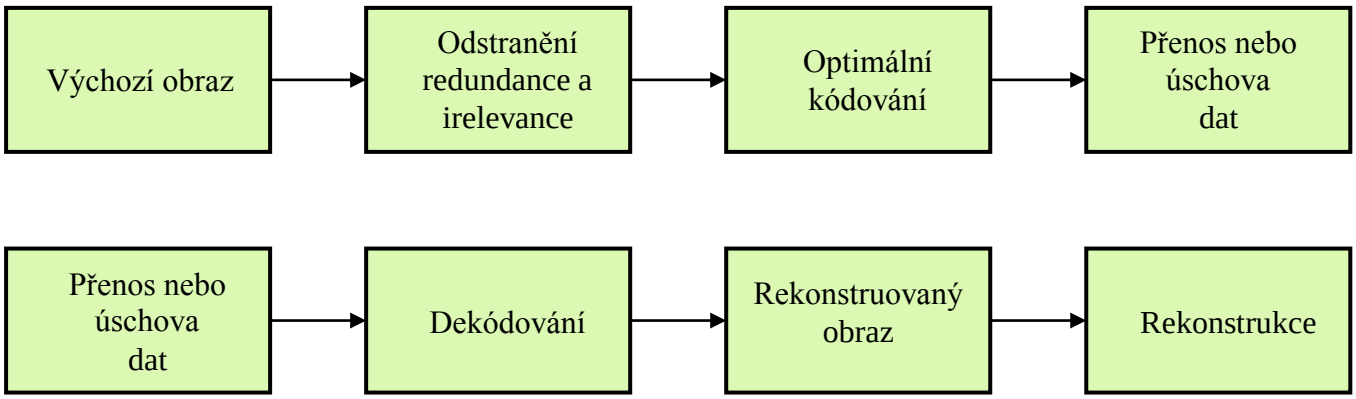
Nejdříve ale než si řekneme něco o těchto standardech, si bychom měli položit otázku, jaké jsou vlastně kladeny nároky na obrazové formáty a tím potažmo i na kompresní techniky, které tyto formáty používají. Vzhledem k tomu, že v dnešní době jsou digitální obrazová data, ať už statická nebo dynamická, používaná v hojné míře a setkáváme se s nimi doslova na každém kroku, jsou kladené nároky enormní. Právě proto, že v posledním desetiletí prodělala digitální obrazová data značný rozvoj, stoupla s ním i jejích popularita a vzrostla oblast použití a uplatnění. Se stále dostupnější digitální technikou, fotoaparáty, scannery, mobilní telefony s vestavěnými fotoaparáty a bezesporu i osobní počítače vzniká z řad běžných uživatelů nárok na snadné použití, úpravu a manipulaci s těmito daty.

Digitálně pořízené obrazy však musíme taky nějak uchovat, ať už pro pozdější úpravu nebo pro samotné prohlížení. Zde nám vznikají nároky na kapacitu přenosového nebo paměťového média. V dnešní době, i přes rostoucí výkon výpočetní techniky a kapacitu přenosových a paměťových médií je potřeba redukovat velikost objemných digitálních obrazových dat, neboť jejich ukládání v nekomprimované podobě, obzvláště při stále vzrůstajícím rozlišení obrazových čipů digitálních fotoaparátů a videokamer, zabírá příliš mnoho prostoru na těchto médiích.

Existuje více druhů komprese. V podstatě ji můžeme rozdělit na ztrátovou a bezztrátovou. U bezeztrátové komprese dostaneme po kódovacím a dekódovacím procesu původní data v nezměněné formě, kdežto ztrátová komprese značně snižuje potřebnou velikost na médiu, ale na úkor ztráty informace, která je už v dekódovacím procesu nevratná. Z toho tedy vyplývá, že u komprese dat se snažíme o to, aby byla velikost dat co nejmenší při stávající kvalitě nebo aby byla kvalita co největší pří zachování fyzické velikosti obrazu.

2 KOMPRESE OBRAZOVÝCH DAT

Účelem komprese dat bývá obvykle snížení objemu dat. Využívá se při kódování téměř všech digitálních informací - obrazu, zvuku apod. Rozlišujeme kompresi dvojího druhu a to **ztrátovou** a **bezztrátovou.** Bezztrátová komprese je taková, kdy po dekódování jsou data v původní podobě. Ztrátová komprese využívá nedokonalostí lidských smyslů, díky tomu může např. z obrazu odstranit části, které lidský oko nedokáže rozeznat.



Obr. 2.1: Obecné schéma komprese a dekomprese obrazu

Pokud bychom se na kompresi podívali z pohledu, jaké části informace odstraňujeme, mohli bychom tyto informace rozdělit na dvě složky a to na **redundantní** a na **irelevantní** složku. Redundantní část představuje tu část informace, kterou, když odstraníme ze signálu, můžeme původní signál zrekonstruovat bez zkreslení. Snažíme se tedy o potlačení informační nadbytečnosti jiným způsobem zápisu, což je zcela reverzibilní proces.

Například signál, který je reprezentován následující řadou vzorků *s(n):*

$$s(n) = \{\ 7,7,7,7,7,7,7,7\ \} \qquad (1.1)$$

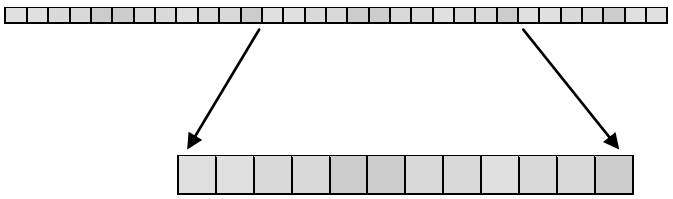
Tuto sekvenci můžeme zapsat podle předem domluveného klíče { *n* – počet vzorků; *i* – hodnota vzorku } jako { 8,7 }. Tímto výrazně kratším zápisem snížíme velikost signálu, ale neztratíme žádné informace, respektive jsme schopni po dekompresi dostat totožný signál.

Zato irelevantní část představuje tu část informace, která je sice postřehnutelná v datovém signálu, ale není postřehnutelná lidskými smysly ve výsledném obrazu. Touto ztrátovou kompresí je možné ušetřit až 95% velikosti obrazu při zachování kvalitního snímku.

Vezměme si například posloupnost 12 vzorků *m(n):*

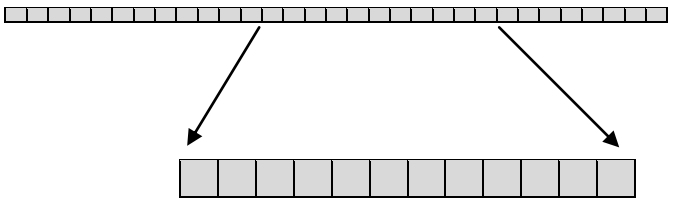
$$m(n) = \{ 63\ 63\ 64\ 64\ 65\ 65\ 64\ 64\ 63\ 64\ 64\ 65 \} \quad (1.2)$$

z nichž každá hodnota představu stupeň šedi jednotlivého obrazového bodu – **pixelu.** Pro představu by to v obrazu vypadalo asi takto:



Obr. 2.2: Náhled na jednotlivé pixely.

Pokavaď bychom ale jednotlivé číselné hodnoty odstínů nahradili aritmetickým průměrem celého řádku, tedy m'(n) = { 64 64 64 64 64 64 64 64 64 64 64 64 } a následně použili zápis, který se používá u komprese Run-Length Encoding, zkráceně RLE, { 12;64 }, dosáhli bychom tímto značné komprese, ale výsledná změna by byla v obrazu o rozlišení 1024x768 pixelů téměř nepostřehnutelná.



Obr. 2.3: Náhled na jednotlivé pixely.

Takto zkomprimovaný obrázek je sice velmi blízký originálu, ale musíme mít na paměti, že jakmile něco ztrátově uložíme, pak už původní obrázek zpět nikdy nezískáme. Je tedy nutné, abychom před uložením obrázku vždy zvážili, na co ho budeme používat a jestli výsledné zkreslení nebude na snímku patrné.

Další dělení komprese je na **symetrickou** a **asymetrickou**. U symetrické trvá komprese i dekomprese stejně dlouho, kdežto u asymetrické tomu tak není. Většinou trvá déle komprese.

3 BEZZTRÁTOVÁ KOMPRESE

Bezztrátová komprese se vyznačuje tím, že původní signál zdroje informací je možné obnovit bez jakékoliv ztráty kvality (původní signál a signál po kompresy a následné dekompresi jsou totožné). Bezztrátovou kompresi používáme především pro situace, kdy potřebujeme zachovat 100% z původního obsahu, ať už je to u obrazu určený pro velkoformátový tisk nebo u lékařských snímků, kde je důležitá každá maličkost.

Jednou z nejjednodušších kompresních metod je již výše zmiňovaná RLU. Její princip je velmi jednoduchý a spočívá v tom, že znaky, které se vyskytují v kódovaném slově vícekrát za sebou, zaznamená tím způsobem, že nejprve zapíše počet opakování a potom znak o který se jedná. Pokud by tedy kódované slovo mělo toto složení: „aaaaaaaaaahhhhhhhooooooojjjjjjjj“, po zakódování by byl zapsán jako: „10a7h7o8j“. Naopak pokud bychom měli kódované slovo: „ahoj“, po zakódování: „1a1h1o1j“ by se slovo nezmenšilo, ale zvětšilo. RLE algoritmy jsou ovšem schopny zabránit zbytečným kompresím a umí rozpoznat, jestli znak označuje počet opakování nebo samotný znak.

Dále se pro bezztrátovou kompresi dat používají především dva typy metod a to statistické metody, které pracují s pravděpodobnostmi výskytů znaků v řetězci, např: Huffmanovo kódování a dále to jsou Slovníkové metody, které vytvářejí indexovaný slovník opakujících se částí kódu. Tuto metodu používá LZ (Lempel-Ziv) komprese a z ní vycházející LZW (Lempel-Ziv-Welch) komprese.

Nevýhodou pro statistické a slovníkové metody je, že jeho schopnost komprimace závisí na charakteru obrazu. U obrazů, které obsahují souvislé plochy stejné barvy, a nebo často opakující se množiny je úspěšnost komprese vysoká a u takovýchto obrazů lze dosáhnout snížení fyzické velikosti až o 90%. Tento způsob komprese se ovšem nehodí na pestrobarevné snímky s velkým množstvím a málo se opakující datové informace.

3.1 HUFFMANOVO KÓDOVÁNÍ

Huffmanovo kódování pracuje na principu nahrazování jednotlivých nosičů informace kódem o určitém počtu bitů a to tak, aby kódové slovo pro nosiče informací s největší pravděpodobností výskytu mělo nejkratší možnou délku. Nosičem informace může být libovolné množství bitů, tedy 1 bit, ale třeba i slovo o délce 8 bytů. Vzhledem k tomu, že pravděpodobnost výskytu jednotlivých znaků v souboru dat je odlišná pro každý soubor, je jasné, že seznam kódových slov se bude měnit.

Pro přesnější pochopení kódování Huffmanovým kódem si sestavíme tabulku, kde budeme předpokládat množinu událostí označené X, které nastávají s určitou pravděpodobností výskytu P(X).

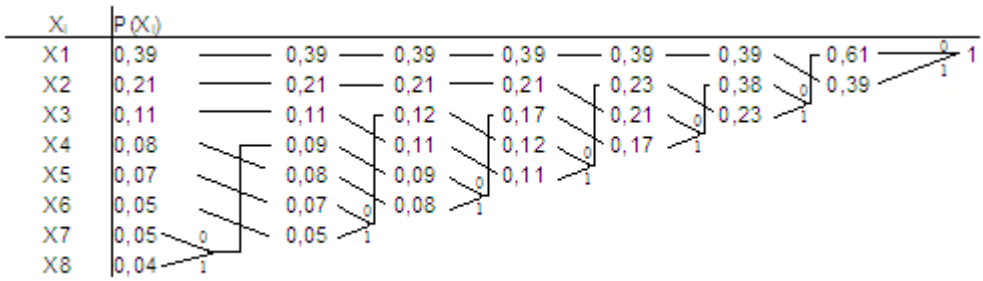
událost X_i	$P(X_i)$
X1	0,39
X2	0,21
X3	0,11
X4	0,08
X5	0,07
X6	0,05
X7	0,05
X8	0,04

Obr. 3.1: Tabulka pravděpodobností.

Princip Huffmanova kódování je jednoduchý a snadno pochopitelný. Nejprve je potřeba zjistit četnost jednotlivých symbolů, tedy jaký je jeho procentuální podíl ze všech symbolů. Znaky se pak podle četnosti seřadí a vždy dva s nejmenší četností se spojí a spojnice se ohodnotí 0 a 1. Postup se opakuje a vzniká tak jakýsi strom. Hrany (cesty) ke znaku pak dávají jeho kódování. Znak s nejvyšší četností se bude kódovat pomocí nejkratší posloupnosti.

Všimněte si, že celkový součet všech pravděpodobností je 1. Při kódování postupujeme takto: Nejdříve sečteme poslední dvě pravděpodobnosti (0,04+0,05=0,11) a vytvoříme nový sloupec pravděpodobností, kde ty dvě, které jsme sčítali, nahradí jejich součet. Všechny pravděpodobnosti v novém sloupci seřadíme sestupně podle velikosti.

Všechny hodnoty pravděpodobností se mezi sloupci pospojují spojnicemi. Spojnice pravděpodobností *P* (X8) a *P* (X7) se sjednotí, ale předtím přiřadíme *P* (X8) bit kódového slova s hodnotou 1 a *P* (X7) bit s hodnotou 0. Takto postupujeme, dokud se součet posledních dvou čísel nerovná 1. Závěrečné kódování každého slova pak probíhá po spojnicích jako "sbírání" zapsaných bitů kódového slova.



Obr. 3.2: Graf Huffmanova kódování.

Jdeme po spojnicích a zapisujeme všechny bity, které po cestě potkáme. Nakonec se celý zápis obrátí odzadu dopředu a výsledkem je kódové slovo pro danou událost. Takže pro X_5 je výsledné kódové slovo "0100".

událost X_i	$P(X_i)$	kód
X_1	0.39	1
X_2	0,21	000
X_3	0,11	011
X_4	0,08	0011
X_5	0,07	0100
X_6	0,05	0101
X_7	0,05	00100
X_8	0,04	00101

Obr. 3.3: Tabulka kódových slov.

Každé kódové slovo je unikátní a nepodobá se žádnému jinému - žádné kódové slovo netvoří předponu jiného, takový kód nazýváme **prefixový**. Díky tomu není nutné používat žádné speciální znaky jako oddělovače.

Kdybychom chtěli tedy zakódovat řetězec ve tvaru:

$$m = X_1, X_4, X_2, X_6, X_7, X_3, X_2 \quad (1.3)$$

výsledné kódové slovo, podle naší tabulky, by bylo ve tvaru:

$$m' = 10011000010100100011000 \quad (1.4)$$

3.2 Algoritmy Lempela, Ziva a Welche

Historie této metody je celkem nedávná (samozřejmě oproti ostatním metodám). V roce 1977 byl tento algoritmus navržen Abrahamem Lempelem a posléze vylepšen v roce 1978 Jacobem Zivem. Dnes se proto můžeme setkat s tímto algoritmem jako **LZ77** nebo **LZ78**. Vývoj však ještě pokračoval, když v roce 1984 byl upraven Terrym Welschem, tento má potom zkratku **LZW.**

Princip metody spočívá v postupném prohledávání celého souboru tak, že vždy část rozdělíme na dvě „okénka“, kde první tvoří historii, kterou prohledáváme, a druhým okénkem se díváme dopředu a hledáme, zda v něm není posloupnost znaků, která se už jednou vyskytuje v okénku historie. Pokud takovou posloupnost najdeme, nahradíme ji uspořádanou dvojicí (o kolik znaků jdu zpět, délka sekvence).

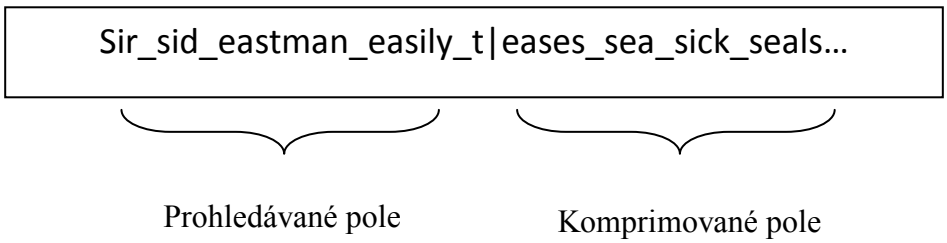
3.2.1 LZ77

Algoritmus LZ77 zveřejnili jeho autoři Lempel a Ziv v roce 1977. Dnes se jedná o celou třídu algoritmů – modifikací původního algoritmu. Algoritmus používá tzv. **posuvné okno**, skrze které nahlíží na vstupní soubor. Tímto oknem přejede postupně přes celý původní soubor. Okno je rozděleno na dvě části: **prohledávané pole** a **komprimované pole**. Když algoritmus najde v komprimovaném poli posloupnost symbolů (řetězec), která se vyskytuje i v prohledávaném poli, do výstupního souboru nekopíruje tuto nalezenou posloupnost, ale zapíše pouze informaci o tom, kde v prohledávaném poli nalezená posloupnost začíná a jak je dlouhá. A na výstup také zkopíruje symbol bezprostředně následující po této (nekopírované) posloupnosti. Do

výstupního souboru jsou tak zapisovány pouze takovéto trojice: pozice kopírované sekvence z prohledávaného pole, délka této sekvence a jeden následující symbol.

Algoritmus začíná tak, že prohledávané pole je prázdné a začátek komprimovaného souboru je v komprimovaném poli. Je tedy zřejmé, že zpočátku se do výstupního souboru zapisují nejčastěji trojice (0,0,X) a efektivita algoritmu je horší, než kdyby nebyla posloupnost komprimována. Avšak po naplnění prohledávaného pole začne být algoritmus úspěšnější. Při hledání co nejdelšího stejného řetězce v obou polích může dojít k tomu, že řetězec v prohledávaném poli „přeteče“ do komprimovaného pole. V trojici, která je zapsaná do výstupu, může být totiž délka kopírovaného řetězce větší, než jeho vzdálenost od hranice mezi prohledávaným a komprimovaným polem. Délka prohledávaného pole se pohybuje nejčastěji v kilobytech, zatímco délka komprimovaného pole řádově v bytech, či desítkách bytů.

Běh algoritmu znázorníme na následujícím příkladu:



Obr. 3.4: Kódové slovo pro LZ77.

Kompresní algoritmus prohledává prohledávané pole pozpátku (zprava doleva) a hledá výskyty symbolu „e“, který je prvním symbolem v komprimovaném poli. Najde symbol „e“ ze slova „easily“. Toto „e se nachází ve vzdálenosti 8 symbolů od konce prohledávaného pole.

Potom algoritmus hledá co nejdelší stejný řetězec začínající symbolem „e“ v obou polích. Tím je řetězec „eas“ o délce 3 symboly. Tyto údaje si algoritmus uloží do pomocných proměnných a pokračuje v hledání symbolu „e“. Další najde ve vzdálenosti 16 symbolů od konce prohledávaného pole. Algoritmus vybírá co nejdelší řetězec a pokud jich je více o stejné délce, vybere ten ve větší vzdálenosti od konce (neboť postupuje zprava doleva). Řetězec začínající symbolem „e“ ve vzdálenosti 16 symbolů od konce má také délku 3, a proto je vybrán (informace o něm je zapsána do

pomocných proměnných). Žádné další výskyty symbolu „e“ v prohledávaném poli nejsou a proto se do zkomprimovaného souboru přidá trojice (16,3,e).

V tomto případě je „e“ symbol bezprostředně následující za zkomprimovaným (nalezeným) řetězcem „eas“. Posléze je posuvné okno je posunuto o 4 symboly dále (3 symboly v komprimovaném poli byly opsány z prohledávaného pole a 1 symbol – „e“ je obsahem do výstupního souboru zapsané trojice). Jestliže by na prvním místě komprimovaného pole byl symbol „w“, nebyl by tento nalezen v prohledávaném poli. Do výstupního souboru by byla zapsána trojice (0,0,w). Posuvné okno by bylo následně posunuto o jeden symbol.

3.2.2 LZ78

Tento kompresní algoritmus, jak už sám název napovídá, byl uveřejněn v roce 1978, vychází z předchozího LZ77. Narozdíl od předchozí verze, tento algoritmus vytváří **dynamický slovník** opakujících se řetězců během komprese, který plní funkci prohledávaného pole z metody LZ77.

Algoritmus postupně rozpoznává a přidává do tabulky (slovníku) vstupní řetězce (řetězce z původního souboru). Do zkomprimovaného souboru zapisuje dvojice: ukazatel do slovníku již dříve nalezených řetězců a po něm bezprostředně následující (v komprimovaném souboru) symbol. První záznam slovníku je dopředu vyhrazen pro prázdný řetězec. Kromě tohoto záznamu je slovník na začátku běhu algoritmu prázdný. Postup při kompresi lze popsat takto:

Do pomocné proměnné *c* je načten první symbol ze vstupního souboru a do pomocného řetězce *W* je uložen prázdný řetězec. Zbytek algoritmu se opakuje až do zpracování celého původního souboru. Algoritmus zjistí, zda se řetězec *W* prodloužený o symbol z proměnné *c* (v dalším kroku budeme tento řetězec označovat jako *W* * *c*) již ve slovníku vyskytuje. Pokud ano, prodlouží se řetězec *W* o symbol *c* (tzn. *W* * *c* je ukládáno do *W*). V opačném případě se do výstupního souboru zapíše dvojce: ukazatel na známý řetězec *W* a symbol *c*. Dále se do slovníku přidá řetězec *W* * *c* a pomocný řetězec *W* je naplněn prázdným řetězcem. Do proměnné *c* je načten další symbol. Cyklus se opakuje.

K vyjádření odkazu na řetězec se v praxi většinou používá 12-ti bitová hodnota. Toto číslo tak udává maximální velikost slovníku. Na počátku je slovník inicializován tak, že první slovníková položka je prázdný řetězec. Když se slovník zaplní, může být vymazán a stavěn znovu. Někdy se pro postupné vymazávání položek ze slovníku používá metoda LRU – Last Recently Used, kdy jsou ze slovníku odstraňovány nejdéle nepoužité řetězce. Dalším možným řešením hrozby přeplnění slovníku je, že se zastaví jeho zvětšování a po obsazení vyhrazené paměti se do něj nepřidávají další záznamy. Důležité je, že vybudovaný slovník nemusí být součástí výstupního souboru. Dekompresní algoritmus totiž dokáže (stejný) slovník budovat podle dat načítaných ze zkomprimovaného souboru.

Běžně používané kompresní programy jako ARJ, ZIP nebo RAR využívají zpravidla některou modifikaci algoritmu LZ78 spolu s následným Huffmanovým kódováním.

3.2.3 LZW

Další metodou, která vychází z předešlých LZ77 a LZ78 je často zmiňovaná metoda LZW. Jedná se o modifikaci navrženou panem Welchem pro hardwarovou kompresi dat při ukládání na pevný disk (a dnes využívanou mimo jiné ve formátu **GIF, TIFF, PNG**). Toto vylepšení spočívá v tom, že slovník je na počátku práce algoritmu inicializován tak, aby obsahoval všechny symboly používané abecedy. Princip běhu algoritmu je následující:

Symboly ze vstupního souboru jsou načítány a akumulují se v pomocném řetězci *W*. Tento proces pokračuje, dokud je obsah *W* totožný s některým ze záznamů ve slovníku. Když se ale obsah řetězce *W* ve slovníku najít nepodaří, je na výstup zapsán ukazatel do slovníku na řetězec *W* zkráceného o poslední znak, do slovníku je přidán řetězec *W*. Následně je *W* naplněno symbolem z *c*. Stejně jako u LZ78, ani zde nemusí být vytvořený slovník součástí zkomprimovaného souboru, protože dekompresní algoritmus je schopen jej sám vytvořit podle dekomprimovaných dat. Drobné vylepšení, se kterým se LZW často používá v praxi, je, že slovník má nejprve 512 položek a ve výstupním souboru tedy stačí pouze 9 bitů pro popis ukazatele do něj. Po jeho naplnění se slovník rozšíří (zdvojnásobí) a musí se tedy používat 10 bitový ukazatel do slovníku. Slovník se zdvojnásobuje pokaždé, když je to třeba, až do

zaplnění vyhrazené paměti. Potom se postupuje některým ze způsobů popsaných u LZ78.

Tyto metody dosahují velmi dobrých kompresních poměrů u téměř všech druhů dat. Proto se využívají velmi často. Používané jsou ke komprimaci GIF nebo TIFF souborů v archivačních programech jako je ARJ nebo ZIP. Velmi důležitá vlastnost je také možnost algoritmu průběžně odesílat komprimovaná data, proto se používá u faxů, modemů a telefonů.

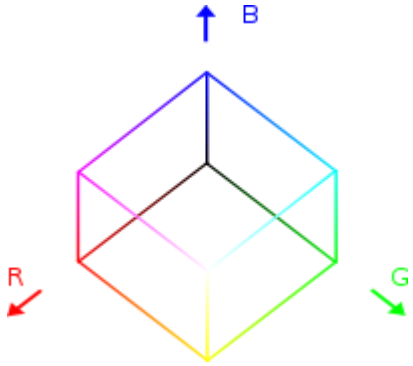
4 ZTRÁTOVÁ KOMPRESE

Postupy pro ztrátovou kompresi obrazu jsou založené na tom, že lidské oko je méně citlivé na barvu než na jas a na tom, že člověk vnímá méně vysoké frekvence obsažené v obrazu než frekvence nízké. Při ztrátové kompresi tedy dochází ke ztrátě těch informací, na které je lidský zrak méně citlivý. Díky těmto ztrátám informací dochází k výraznému zmenšení zkomprimovaných dat.

Dále tedy transformace nebo přeskupení dat je uzpůsobeno tak, aby bylo jednoduše možné oddělit důležité informace, na které je lidské vidění citlivé, od méně důležitých. Méně důležité informace jsou pak uloženy s menší důležitostním koeficientem nebo vůbec. Tím dochází ke ztrátě informací a ke zmenšení objemu výsledných dat. Data jsou dále zpracována některou bezeztrátovou metodou (např. pomocí Huffmanova kódování).

4.1 BAREVNÉ MODELY

Nejdříve si tedy řekneme něco o barevných modelech. Každá barva je udána mohutností tří základních barev – složek (červené - red, zelené – green a modré – blue, odtud RGB). Základní barvy mají vlnové délky 630, 530 a 450 nm. Mohutnost se udává buď v procentech, nebo podle použité barevné hloubky jako určitý počet bitů vyhrazených pro barevnou složku (pro 8 bitů na složku je rozsah hodnot 0 – 255, pro 16 bitů na složku je rozsah hodnot 0 – 65535), přičemž čím větší je mohutnost, tím s vyšší intenzitou se barva složky zobrazuje.



Obr. 4.1: Barevná jednotková krychle.

Model RGB je možné zobrazit jako krychli, ve které každá z kolmých hran udává škálu mohutností barevných složek. Potom libovolný bod se souřadnicemi (r, g, b) v této krychli udává hodnotu výsledné barvy.

Následující tabulky nám zobrazují podíl složek R, G, B na výsledné barvě v 8-mi bitové barevné hloubce.

R	G	B	Barva
255	0	0	Červená
0	255	0	Zelená
0	0	255	Modrá
255	255	255	Bílá

Obr. 4.2: Tabulka RGB.

Obrácený systém je subtraktivní systém CMYK, kdy pro každou jeho barvu (mino černé) je použita směs dvou základních barev RGB s maximální mohutností.

R	G	B	Barva
0	0	0	Černá
255	255	0	Žlutá
255	0	255	Purpurová
0	255	255	Azurová

Obr. 4.3: Tabulka CMYK.

4.2 KONVERZE BAREVNÉHO MODELU

Barevný model RGB se ke ztrátové kompresi obrazu příliš nehodí, protože hodnoty jednotlivých složek R, G, B se mezi sousedními pixely často mění. Změna jedné hodnoty navíc ovlivní nejen barevnou složku, ale také jas bodu a lidský zrak nejvíce vnímá právě jas. Z tohoto důvodu je vhodnější pixel reprezentovat jako jas (budeme ho označovat parametrem Y) a další složky, které udají vlastní barvu (označované jako C_b a C_r). U složek udávajících barvu je pak při kompresi možné dopustit se větší nepřesnosti než u jasu, protože to lidský zrak hůře rozpozná. Tímto převodem na jiný barevný model dosáhneme větší komprese.

Nejrozšířenější varianta barevného modelu je varianta označovaná jako JPEG-$Y'C_bC_r$, definována ve specifikaci JFIF. V této specifikaci jsou uvedeny rovnice pro převod barvy z modelu RGB na dané $Y'C_bC_r$, a zpět. Tyto rovnice očekávají 8 bitové složky R, G, B (hodnoty v intervalu 0 až 255) a produkují taktéž 8 bitové složky $Y'C_bC_r$ v tomtéž intervalu.

Převod z modelu RGB na model $Y'C_bC_r$:

$$\begin{aligned} Y' &= 0.299 * R + 0.587 * G + 0.114 * B \\ C_b &= -0.1687 * R - 0.3313 * G + 0.5 * B + 128 \\ C_r &= 0.5 * R - 0.4187 * G - 0.0813 * B + 128 \end{aligned} \tag{1.5}$$

Převod z modelu $Y'C_bC_r$ na RGB:

$$\begin{aligned} R &= Y' + 1.402 * (C_r - 128) \\ G &= Y' - 0.34414 * (C_b - 128) - 0.71414 * (C_r - 128) \\ B &= Y' + 1.772 * (C_b - 128) \end{aligned} \tag{1.6}$$

Po tomto převodu se výsledky zaokrouhlí na nejbližší celá čísla, a tedy tímto dochází již ke ztrátám. Výsledkem převodu obrazu z RGB do $Y'C_bC_r$ jsou 3 samostatné složky, které je výhodnější dále komprimovat odděleně. Intenzita Y´ se často komprimuje

jinou metodou než C_b a C_r, protože lidské oko je více citlivé na jas než na barvu. Složky C_b a C_r, se mění velmi mírně a jsou blízké nule, tím pádem jsou výhodné pro kompresi.

4.3 POUŽÍVANÉ ALGORITMY PRO ZTRÁTOVOU KOMPRESY OBRAZU

Zmiňované metody LZW a Huffmanovo kódování se na obrázky s častými barevnými přechody moc nehodí. Tato metody se spíše hodí na kódování fotografii, kde převládají sousedící pixely s malou nebo žádnou barevnou odlišností nebo textové obrázky.

U diskrétní kosinovy transformace je kompresní poměr řízen požadavkem na výši kvality dekomprimovaného obrazu a naopak vyniká velkou kompresí u obrázku s častými barevnými přechody.

4.3.1 DISKRÉTNÍ KOSINOVA TRANSFORMACE

Diskrétní kosinová transformace, z anglického překladu (Discrete Cosine Transform, zkratka DCT) je integrální transformace často používaná při komprimaci obrazu. Tato transformace je odvozena od Fourierovy transformace (DFT) a stejně jako ona převádí signál, v našem případě dvourozměrný signál, do spektrálních složek (koeficientů), ale na rozdíl od ní produkuje pouze reálné koeficienty.

Existuje několik typů DCT a několik metod jejich výpočtu. My se budeme zabývat transformací označovanou jako DCT2 tedy dvojrozměrnou DCT.

$$F(u,v) = \alpha_u \alpha_v \sum_{i=0}^{M-1} \sum_{\mathrm{j}=0}^{N-1} f(i,j) \cos\frac{\pi(2i+1)u}{2M} \cos\frac{\pi(2j+1)v}{2N} \qquad (1.7)$$

a inverzní iDCT2 je definována jako:

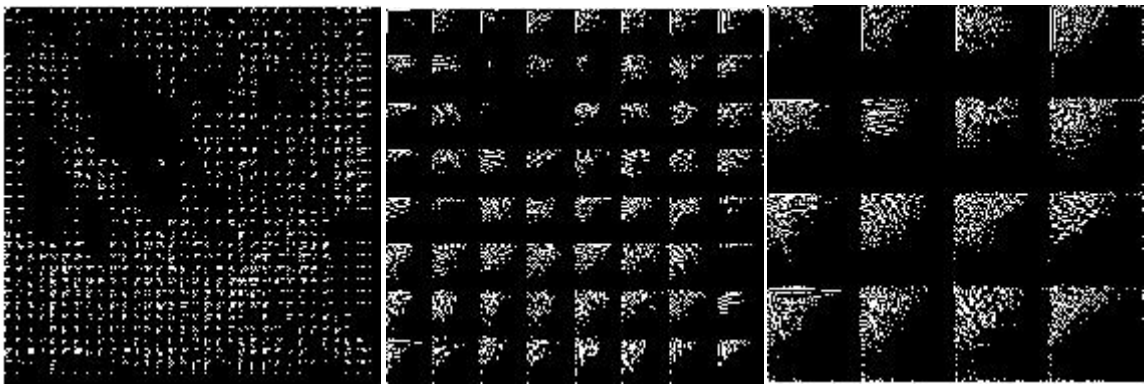
$$f(i,j) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha_u \alpha_v F(u,v) \cos\frac{\pi(2i+1)u}{2M} \cos\frac{\pi(2j+1)v}{2N} \qquad (1.8)$$

Spektrum DCT se skládá ze spektrálních koeficientů, kterých je stejně jako vstupních obrazových bodů, pokud má tedy vstupní obraz rozlišení např. 256x256 obrazových bodů, získáme po aplikaci DCT 256x256 spektrálních koeficientů.

Obr. 4.4: Obrázek spektrálních koeficientů.

U dvourozměrné verze se vyprodukuje matice koeficientů. Nejdůležitější koeficient s indexem *(0, 0),* vlevo nahoře, vyjadřuje stejnosměrnou složku. Další a méně významné koeficienty vzdalující se od tohoto bodu horizontálně či vertikálně vyjadřují obsah bázové funkce s odpovídajícími horizontálně či vertikálně znásobenými frekvencemi, na kterých lze uplatnit větší ztrátu informace.

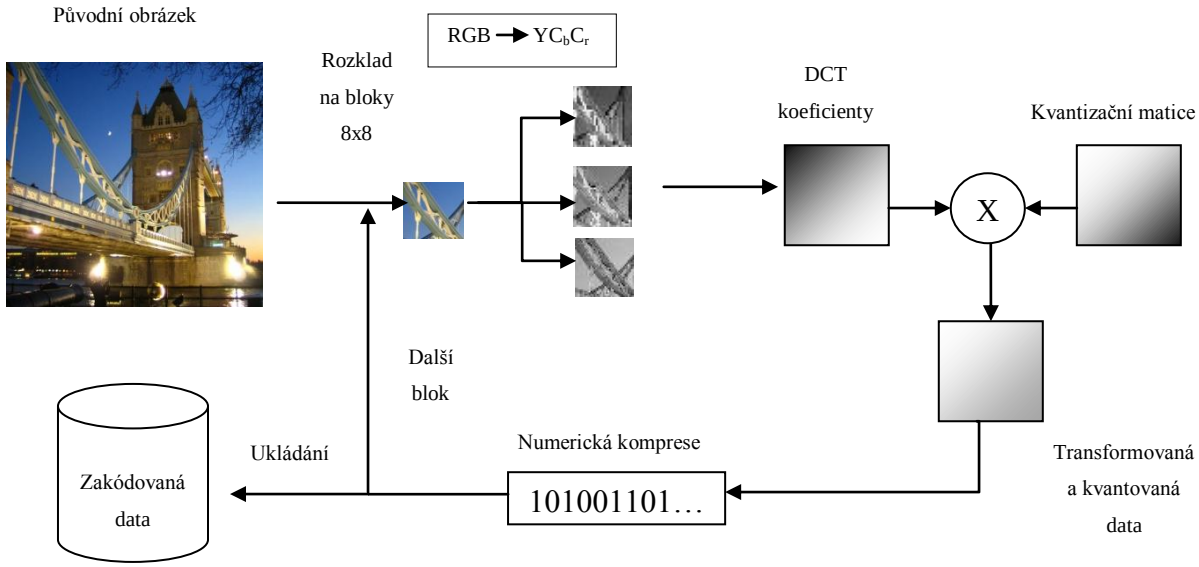
Jestliže by se DCT transformace počítala z celého obrázku najednou, potom by zároveň spolu s rostoucím rozlišením obrázku prudce stoupal i čas potřebný na transformaci. Kvůli tomu se provádí transformace po dostatečně velkých na sobě nezávislých blocích. Nejběžněji se používají bloky o rozměrech 8x8 bodů, jen zřídka bloky o rozměrech 16x16, 32x32 a 64x64 obrazových bodů a to kvůli větší časové náročnosti.



Obr. 4.5: Obrázky rozdělení na bloky zleva 8x8, 32x32, 64x64.

DTC komprese je založena na redukci irelevance prováděné ve frekvenční oblasti. Z toho plyne, že se jedná o kompresi ztrátovou. Nejvíce informací je uloženo v nízkofrekvenčních složkách, a proto se můžeme na vysokofrekvenčních složkách dopustit větší ztráty a kvantovat je tak hruběji než koeficienty nízkofrekvenční. Na začátku se odčítá od koeficientů číslo 128, kvůli snížení stejnosměrné složky. Při rekonstrukci obrazu se toto číslo opět přičte. Každý segment, obsahující nejčastěji 64 koeficientů, se podělí vhodně zvoleným číslem z kvantizační matice, a protože je matice sestavena tak, aby co nejvíce vyhovovala vlastnostem lidského zraku, jsou nízké frekvence děleny nízkým číslem, kdežto vysoké frekvence vysokým číslem.

Navolením kvality výsledné komprimace volíme tedy i vhodnou kvantizační matici, která nám určí, kolik koeficientů bude nulových nebo blízkých nule a kolik koeficientů nám ponese data. Pokavaď si zvolíme vysokou kvalitu komprese, zvolíme tím i kvantizační matici, která nám po dělení zanechá více nenulových koeficientů, ale zároveň s tím se nám i zvýší nároky na potřebný paměťový prostor a naopak. Takto získané koeficienty se následně zaokrouhlí na nejbližší celé číslo, z čehož dostaneme většinu vysokofrekvenčních koeficientů s nulovou hodnotou. Spektrální koeficienty se poté zakódují nejčastěji entropickým kódováním do výstupního souboru. K obnovení obrázku je zapotřebí inverzního postupu.



Obr. 4.6: **Zjednodušené schéma DCT komprese obrazu.**

4.3.2 DISKRÉTNÍ VLNKOVÁ TRANSFORMACE

Vlnková transformace vznikla z důsledku snahy získat časově frekvenční popis signálu. Fourierova transformace umí poskytnout pouze informaci o tom, která frekvence obsahuje signál, ale neříká už nic o jejím umístění v čase. Nožným řešením je použít STFT – krátkou fourierovu transformaci, která řeší problém tak, že posunuje okno v čase a vymezuje tak krátký úsek signálu, ze kterého se počítá spektrum. Jenže u této metody nelze současně určit frekvenci i její polohu. U Krátkých časových oken získáme dobré rozlišení v čase, ale špatné frekvenční rozlišení a u dlouhých časových oken získáme dobré frekvenční rozlišení, ale naopak špatné časové rozlišení. Proto při vlnkové transformaci se používají okna s vhodnou změnou šířky a díky tomu jsme schopni dosáhnout optimálního rozlišení v čase i frekvenci zároveň.

Obr. 4.7: **Obr. Velikosti oken pro transformace.**

Vlnková transformace je definovaná jako konvoluce mateřské vlnky a vstupního signálu.

Konvoluce dvou diskrétních signálů se značí * a je definovaná jako:

$$(f * g)[m] \sum_{n} f[n] * g[m-n] \tag{1.9}$$

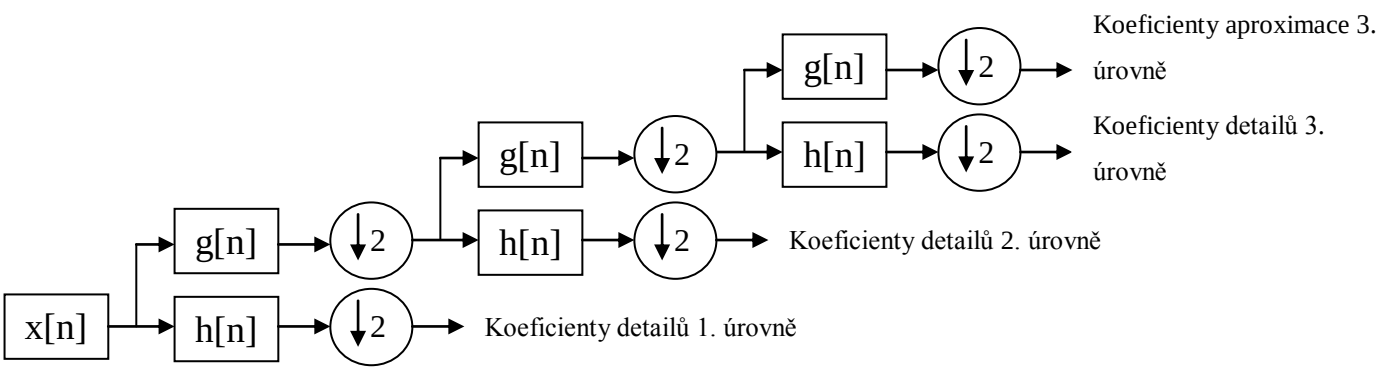
Waveletovu transformaci pomyslného signálu *x* lze chápat jako průchod skrze několik filtrů. Signál projde nejprve dolní propustí s impulzní odezvou g a současně i horní propustí s impulzní odezvou h. Na výstupu těchto filtrů dostaneme dva signály s odlišnými vlastnostmi. Horní propust poskytuje koeficienty detailů (vysoké frekvence) a dolní propust poskytuje koeficienty aproximace. Tyto filtry jsou popsány následujícími rovnicemi:

$$(y_{dolní})[n] = \sum_{k=-\infty}^{\infty} x[k] * g[2*n-k] \tag{2.0}$$

$$(y_{horní})[n] = \sum_{k=-\infty}^{\infty} x[k] * h[2*n-k] \tag{2.1}$$

Průchodem přes tyto filtry se zmenší frekvenční pásmo obou signálů na poloviny a díky tomu můžeme oba nové signály podvzorkovat dvěma.

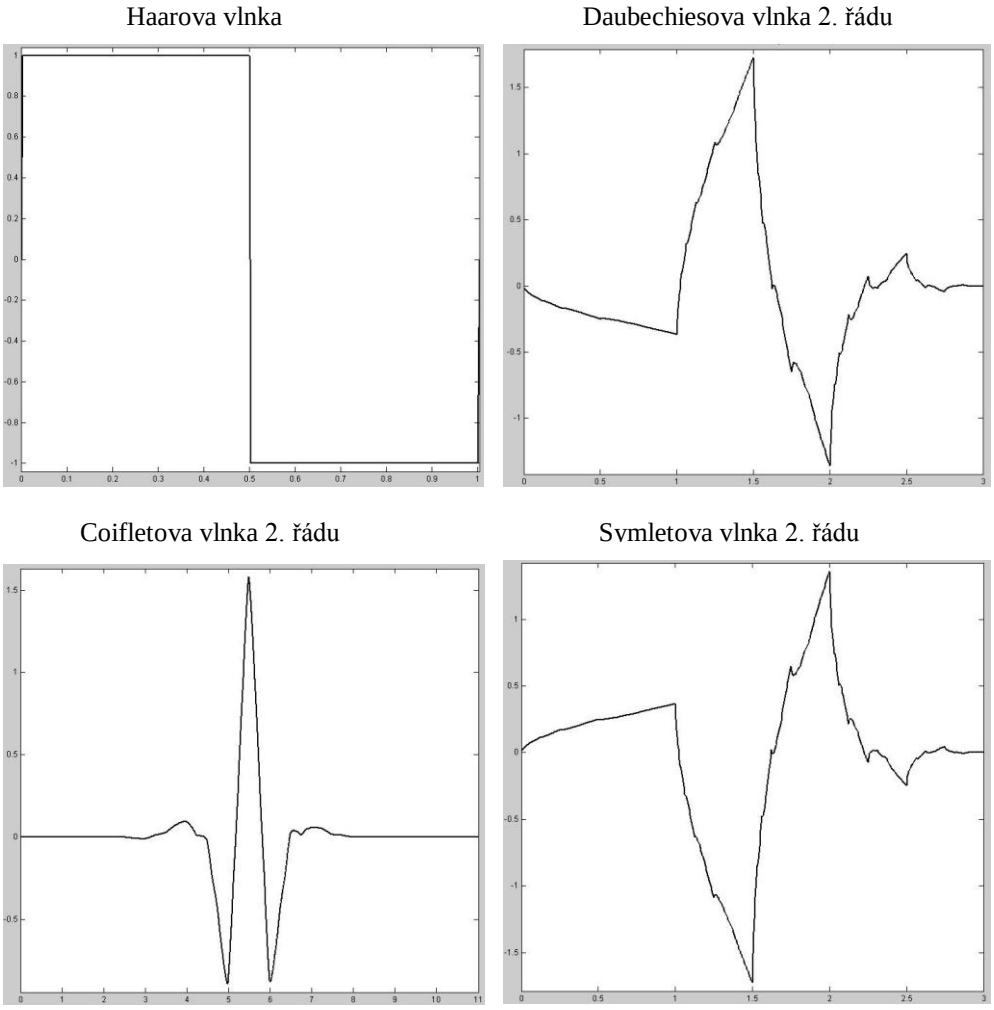
Při dekompozici signál prochází skrz kvadraturní zrcadlo a dále je tento signál decimovám. Po každém takovémto procesu dostaneme na výstupu 2 soubory koeficientů, koeficienty detailů s použitím horní propusti (HP) a koeficienty aproximace s použitím dolní propusti (DP). Výstup DP lze dále analyzovat opakovaným rozkladem až do vyčerpání vstupní frekvence.



Obr. 4.8: Rozložení na koeficienty.

Na každém stupni v diagramu je signál rozložen do nízkých a vysokých frekvencí. Kvůli procesu dekompozice musí být vstupní signál násobkem 2^n, kde n je počet stupňů dekompozice. Pokud má tedy signál délku $N=2^k$ a délka dekompozice je P, pak dostaneme $N/2^{-1}+ N/2^{-2}+\ldots..+N/2^{-p+1}+ N/2^{-p}$ koeficientů detailu a $N/2^{-p}$ koeficientů aproximace.

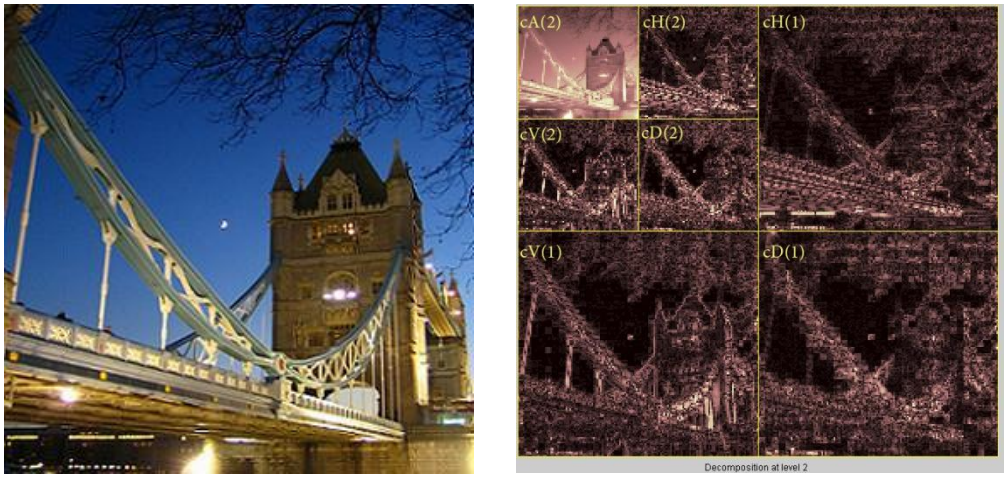
Výchozí formulace vlnky nám určuje filtry horní a dolní propusti. Kvůli definici vztahu pro kvadraturní zrcadlo můžeme použít pouze vlnky, které této definici vyhovují. V praxi se používají Haarova vlnka, Coifletova vlnka a jiné od těchto odvozené.



Obr. 4.9: Různé typy vlnek

Podobně jako dekompozici jednorozměrného signálu se lze představit dekompozici obrazu pomocí průchodu dvourozměrnou bankou. U vektorové kvantizace snažíme nalézt podobnosti mezi jednotlivými prvky a využít je. Jestliže se

pozorně podíváme na dekompoziční obrazec a na rozložení vlnkových koeficientů, zjistíme jistou podobnost mezi stejnými skupinami koeficientů v různých kontovacích úrovních. Můžeme také pozorovat, že s rostoucí úrovní se vždy koeficienty rozměrově dvakrát zmenší. Z toho vyplývá, že každému koeficientu v určité úrovni odpovídá skupina čtyř koeficientů v úrovni o stupeň vyšší.



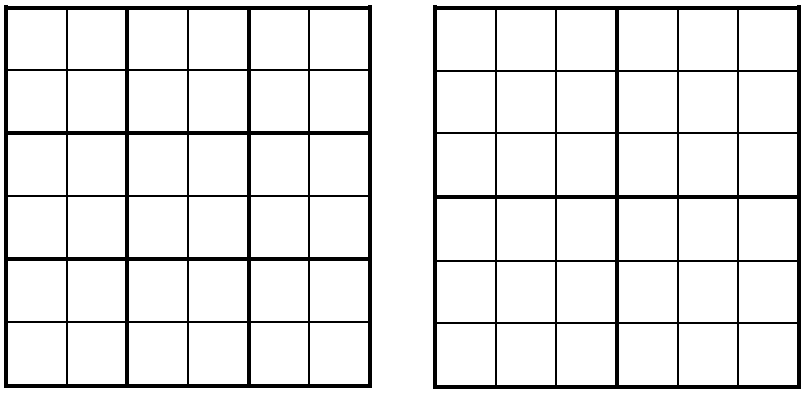
Obr. 4.10: Vlnková transformace 2. Úrovně.

Komprese, stejně tak i jako v případě formátu JPEG, se skládá ze ztrátové a bezeztrátové části. Bezeztrátovou část dostaneme pomocí entropického kodéru. Ztrátová část spočívá v tom, že jednotlivé matice detailů a aproximací jsou kvantovány maticí o rozdílném počtu bitů podle důležitosti. Tuto ztrátovost si můžeme dovolit díky nedokonalosti lidského zraku a tomu, že oko není stejně citlivé na všechny frekvence stejně. Z tohoto důvodu se nízkofrekvenčním koeficientům přiděluje vyšší počet bitů než vysokofrekvenčním koeficientům.

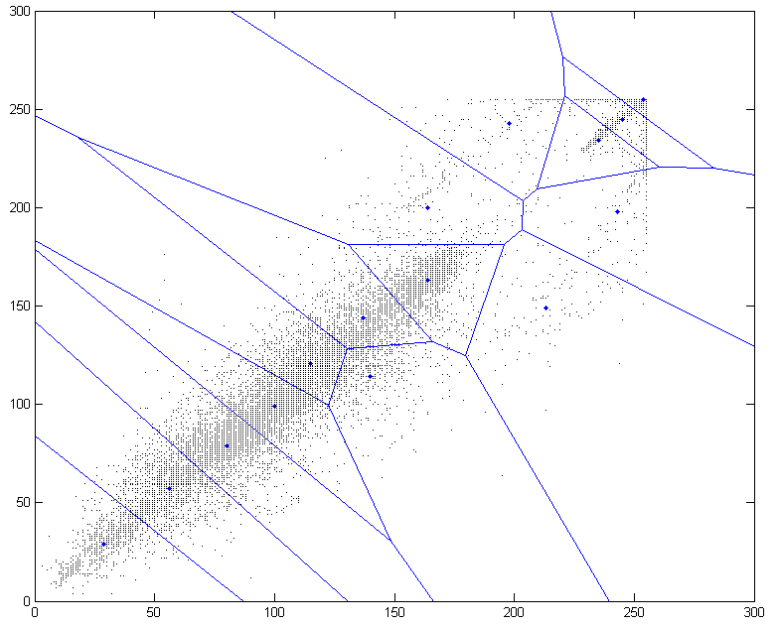
4.3.3 VEKTOROVÁ KVANTIZACE

Tato kompresní metoda, na rozdíl od kosinové transformace, využívá kvantizace ne jednoho, ale hned několika obrazových bodů najednou a to nejčastěji v blokách o velikosti 1x2, 2x2, 3x3, 4x4 a 5x5. V této práci budeme z časového hlediska kvůli náročnosti zpracovaní aplikací používat první tři zmíněné. Spektrální koeficienty z těchto bloků kvantujeme každý jinou hrubostí a to podle toho, jak je lidské oko citlivé na danou frekvenci. Tímto kriteriem je docílena maximální komprese při

minimální ztrátě důležitých dat. Každý takovýto blok bodů tvoří n-tici, tedy vektor. Jedná se tedy o skalární kvantizaci. Pro zjednodušení se následně každá skupina vektorů nahradí reprezentantem a to tak, aby co nejlépe nahradil hodnotu těchto vektorů. Tyto reprezentanty hledáme podle předem zvolených kriterií, např. minimalizace střední kvadratické chyby a vyhledávání se nejčastěji používá adaptivní metoda vyhledávání, jejichž výhoda je v optimalizaci pro zvolený obrázek.



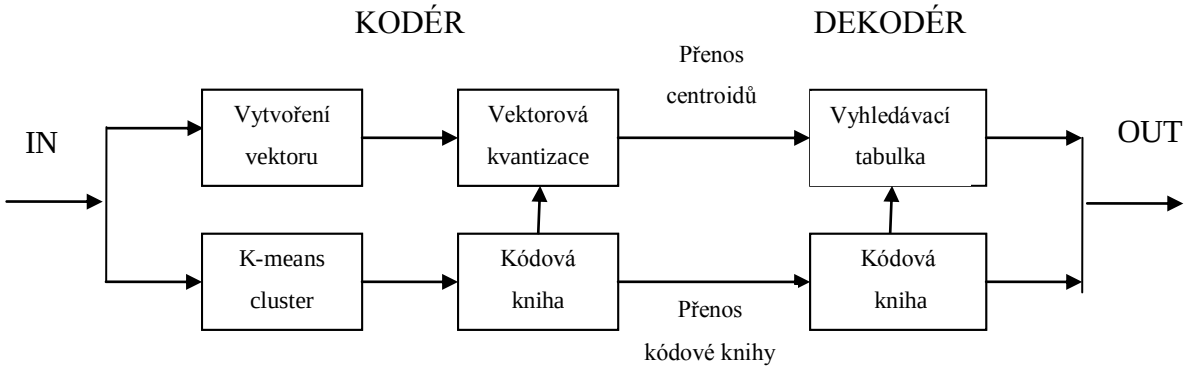
Obr. 4.11: **Bloky o velikosti 2x2 a 3x3 bodů.**



Obr. 4.12: **Voronoiho diagram pro obr. Abbey 1x2.**

Reprezentant nahrazující skupinu bodů nazýváme centroid a množina všech centroidů tvoří kódovou knihu. Hranice mezi oblastmi patřící jednotlivým centroidům se jmenují Voronoiho buňky a vymezují oblasti vektorů, které budou stejně

kvantovány a jsou nahrazeny 1 reprezentantem této oblasti. Na obrázku 3.0 je vyobrazen Voronoiho diagram rozložení 15 optimálních reprezentantů (centroidů) pro obr. Abbey kvantován po blocích 1x2. A zde se dostáváme k největšímu problému VK. Nalezení vhodné kódové knihy pro konkrétní obrázek je velice výpočetně náročné a to i pro moderní počítače. To je důvod, proč budeme používat pouze bloky bodů o velikosti 1x2, 2x2 a 3x3. K jejímu nalezení budeme používat algoritmus K-means clustering. Po této náročné výpočetní operaci následuje vlastní kódováví, což je naopak operace již velmi snadná a nijak nenáročná. Nahrazením vstupních vektorů nejbližšími centoidy dochází k vektorové kvantizaci, ale vzniká zde taky kvantizační šum, který je úměrný rozdílem vstupního vektoru a centroidu. Z toho tedy vyplývá, že čím větší zvolíme kódovou knihu a tím i počet centroidů, tím bude kvantizační šum menší a kvalita komprimovaného obrázku větší. Ke kompresi nám tedy dochází při záměně jednotlivých vektorů za centroidy, které jsou indexovány. Ovšem spolu s reprezentativními body musíme přenášet i samotnou kódovou knihu, která ale není komprimována a tak její velikost musíme volit obezřetně, protože s její rostoucí velikostí nám úměrně klesá výsledný kompresní poměr.



Obr. 4.13: Schéma VK kodéru a dekodéru.

Pro srovnání můžeme vidět na následujících obrázcích, jak dopadla VK s patnácti prvky kódové knihy použitou na obrázek Abbey s různými velikostmi bloků, po kterých se obrázek kvantuje. Na prvním snímku je původní obrázek a dále následují obrázky komprimované po blokách 1x2, 2x2 a 3x3. Na snímcích je zřetelně vidět značné zhoršení kvality s rostoucími velikostmi bloků.

Obr. 4.14: a) Původní obr. Abbey, b) Abbey blok 1x2 PSNR: 29,1 c) Abbey blok 2x2 PSNR: 27,2 d) Abbey blok 3x3 PSNR: 24,5.

Tento jev je dán tím, že VK pro kódování používá palety barev a proto, když chceme kódovat obrázek po větších blocích, je potřeba zvětšit i velikost kódové knihy a tím tedy i počet reprezentantů pro zachováni kvality snímku. Zde je tedy vidět, že pokavaď chceme zachovat kvalitu obrázku, musíme s rostoucí velikostí bloků zvyšovat i velikost kódové knihy, která se ale nekomprimuje. Ovšem zvyšuje se i výsledná velikost objektu a tím úměrně klesá kompresní poměr. Protichůdným argumentem je to, že čím větší je velikost vektoru, tím lépe dosáhneme vyššího kompresního poměru u zakódovaného vektoru o velikosti n x n bodů jedním číslem. Vhodným nastavením těchto dvou parametrů dosáhneme tedy ideálního vyvážení těchto požadavků a tím i kompresního poměru a kvality obrazu.

Pokavaď bychom chtěli zjistit velikost nekomprimovaného obrázku, v tomto případě obrázek Abbey o velikosti 150 x 200 (x x y) obrazových bodů s osmibitovou barevnou hloubkou, spočítá se pomocí jednoduché následující rovnice:

$$S_{nekom} = x * y * 8 \text{ [bitů]}$$

U vektorové kvantizace však musíme počítat s tím, že s vlastními obrazovými daty, tedy indexy jednotlivých vektorů I_n se přenáší i nekomprimovaná kódová kniha C_b, jejichž velikost lineárně roste s počtem reprezentantů v kódové knize.

Velikost komprimovaného obrázku s vektory o velikosti a x b bodů vypočítám pomocí rovnice:

$$S_{kom} = I_n + C_b = \frac{x * y}{a * b} \log_2 k + 8ab \text{ [bitů]}$$

Výsledný kompresní poměr je potom dán jako:

$$C_r = S_{nekom} / S_{kom}$$

4.4 FORMÁT JPEG

Tento zřejmě nejznámější a nejpoužívanější komprimační formát vůbec využívá již výše zmíněnou DCT transformace. Skutečným názvem typu souboru je JFIF, z anglického *JPEG File Interchange Format*. Pod zkratkou JPEG se nám tedy skrývá slova *Joint Photographic Experts Group*, konsorcia, které tuto kompresi navrhlo. Jeho nejčastější přípony jsou: *.JPEG, .JPG, .JFIF a .JPE.* Nejlepších komprimačních poměrů dosáhneme na pestrobarevných obrázcích s častými přechody barev, naopak je nevhodný pro komprimaci textu, perokreseb nebo černobílých ikon, neboť v takovýchto obrazech jsou velmi viditelné artefakty, které tato metoda vytváří při velké kompresi.

Celé kódování probíhá v několika krocích. Nejdříve je objekt rozdělen na segmenty, nejčastěji o velikosti 8x8 pixelů. V této části vzniká největší problém všech kompresí

založených na DCT, protože segmentace je při vysokých stupních komprese velmi viditelná a vnáší do dekódovaného obrazu zkreslení v podobě blokových artefaktů.

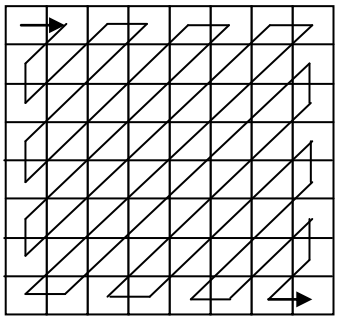
Obr. 4.15: Zobrazení blokových artefaktů.

Po tomto rozdělení následuje převod barevného modelu z RGB na $Y'C_bC_r$. Následuje kvantování, které využívá statické kvantizační matice, kterou se koeficienty dělí na jednotlivé prvky a následně zaokrouhlí. Kvantizační matice přímo svými hodnotami určuje, který koeficient je pro další zpracování důležitý a který ne. Skupina při vytváření formátu JPEG optimalizovala kvantizační matici lidskému zraku tak, aby poskytla maximální možnou redukci irelevantní složky při zachování co nejvyšší kvality. Optimální maticí, označovanou jako Q_{50}, je matice zachovávající přijatelnou kvalitu při vysokém stupni komprese. Stupeň komprese (Sk) se nejčastěji udává jako číslo od 1 do 99. Matice Q_{50} je tedy pro S_k=50. Ostatní matice jsou odvozeny podle vztahu:

Pro Sk > 50 $$Q_{Sk} = Q_{50} * \left[\frac{(100-Sk)}{50}\right] \quad (2.2)$$

Pro Sk < 50 $$Q_{Sk} = Q_{50} * \left(\frac{50}{Ks}\right) \quad (2.3)$$

V případě segmentu o velikosti 8x8 pixelů se všech 64 koeficientů načte do zásobníku podle stanovené „zig-zag“ cesty, která bere v úvahu váhu jednotlivých koeficientů v každém segmentu tím, že nejdříve načte prvek s největší důležitostí a postupuje k méně důležitým prvkům.



Obr. 4.16: Zig-zag cesta.

V případě barevných obrázkům kodér obdobný, jen se transformace provádí odděleně pro jasovou složku Y' *a* barevné složky C_b a C_r. Tyto složky se ze vstupního RGB signálu získají podle vztahu uvedený v kapitole 4.2. Na závěr je použita numerická komprese, která musí zohlednit to, kde se vyskytuje největší počet relevantních koeficientů a také nás zbaví nulových koeficientů, které pro nás nejsou důležité.

Na následujících snímcích můžete porovnat kvalitu a velikost jednotlivých obrázků.

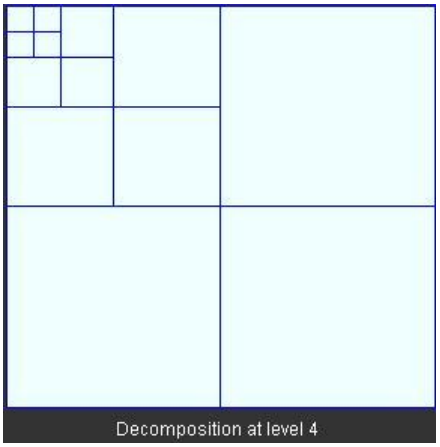
Obr. 4.17: a) Originál – 106 kB b) Sk=50 – 17,5 kB c) Sk=30 – 11,5kB. d) Sk=20 – 9,7 kB e) Sk=10 – 6,84 kB f) Sk=5 – 5,74kB.

4.5 FORMÁT JPEG 2000

Novodobým následovníkem formátu JPEG je bezesporu JPEG 2000. První verze vyšla, jak už jméno napovídá, v roce 2000. Na rozdíl od JPEGu novější verze je vystavěna na diskrétní vlnkové transformaci a od toho se také odvíjejí její vlastnosti. Nový formát ovšem nebyl vyvinut kvůli vysoké efektivnosti kompresního algoritmu, ale šlo o snahu vytvořit univerzální formát ovládající jak ztrátovou, tak i bezeztrátovou kompresi s podporou široké škály vlastností. Předními vlastnostmi formátu JPEG 2000 jsou:

- kompresní algoritmus účinný i pro velké kompresní poměry.
- možnost zpracovávat obrázky větší ne 64000x64000 pixelu.
- možnost komprese nejen přirozených obrázků, ale také šedotónových, počítačové grafiky a i medicínské grafiky.
- vyšší účinnost komprese při zachování stejné kvality obrazu oproti standardu JPEG, zhruba o 20 až 30 procent ve ztrátové kompresi.
- postupný, progresivní, přenos se zvyšující se kvalitou, rozlišením, dle komponent nebo dle prostorového rozprostření.
- ztrátová a bezeztrátová komprese.
- náhodný (prostorový) přístup do bitového toku.
- posun a přiblížení v obraze (*pan, zoom*) s dekompresí pouze nutných částí.
- možnost využití různých barevných módů.
- možnost definování oblastí zájmu, které potom budou komprimovány v lepší kvalitě, popřípadě i s větším rozlišením.
- vysoká odolnost vůči chybám.
- možnost ukládání metadat.

Následně si ukážeme, jakým způsobem JPEG 2000 pracuje. První procedurou je tzv. dlaždicování (tiling, tile - dlaždice). Jedná se o rozdělení obrázku na nepřekrývající se části o stejné velikosti, které jsou dále zpracovávány nezávisle na sobě. Tato procedura má za následek snižující se nároky na operační paměť počítače, ale také umožňuje dekódovat pouze vybranou část obrázku.



Obr. 4.18: 4. stupeň dlaždicování.

Následující částí je transformace barev. Standart JPEG 2000 umožňuje převod jak ztrátový z RGB na *Y'CbCr* nebo bezztrátový RCT (Reversible Component Transformation) do barevného prostoru *YUV*. Následující dvě tabulky zobrazují vztahy transformace:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0{,}25 & 0{,}5 & 0{,}25 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0{,}75 & -0{,}25 \\ 1 & -0{,}25 & -0{,}25 \\ 1 & -0{,}25 & 0{,}75 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \qquad (2.4)$$

Po transformaci barev následuje diskrétní waveletova transformace a to ve dvou možnostech. Nevratná transformace zanáší do výpočtu chybu způsobenou zaokrouhlováním při výpočtu a nebo vratnou transformaci, která při výpočtech pracuje s čísly typu integer s použitím transformace barev do *YUV*. Následuje kvantizace a uspořádání bitových rovin, což je největší předností JPEGu 2000. Toto uspořádání předpřipraví data tak, že se zakódují aritmetickým kódováním a těmto datům lze později přistupovat odděleně, což má za následek schopnost rekonstrukce s kvalitou podle potřeby. Tímto tříděním dat je zajištěna možnost přistupovat k datům jednotlivě. Po kvantizaci je každá matice koeficientů rozdělena na čtvercové nepřekrývající se oblasti. Všechny bity všech koeficientů v bloku jsou kódovány pomocí EBCOT (Embledded Bitplane Coding with Optimal Truncation). Je to proces kódování koeficientů v bloku podle významnosti. Bity se z rovin načítají po čtyřřádkových blocích a v rámci bloku po čtyřbitových sloupcích. Takto získané bity se kódují pomocí MQ-kodéru. Toto kódování je prováděno obsahově závislým binárním aritmetickým kódováním. Závislost je tvořena

stavem devíti okolních sousedů v kódovém bloku. Je nutné dodat, že toto je pouze stručný přehled toho, jakým způsobem pracuje formát JPEG 2000, a vylíčil jsem zde pouze nejdůležitější informace.

I přes nesporné kvality tohoto formátu není tento standart příliš rozšířený a to ze dvou důvodů. Prvním je již jeho zmiňovaný předchůdce JPEG, který je natolik rozšířený a zavedený, že se jen stěží nahrazuje jiným formátem a druhým důvodem je skutečnost, že standart JPEG 2000 je komerční a tudíž se musí za jeho používání platit.

Na následujících snímcích můžete porovnat oba představené formáty:

Obr. 4.19: Srovnáni JPEG a JPEG 2000. a) Sk=5 – 5,74kB b) Sk=5 – 4,86kB.

5 KRITÉRIA HODNOCENÍ OBRAZU

5.1 SUBJEKTIVNÍ KRITÉRIA

Pokavaď nechceme využívat pouze bezztrátovou kompresi, která je založena na redukci redundantních (nadbytečných) informací, ale chceme docílit většího kompresního poměru, budeme muset použít některou ze ztrátových metod komprese obrazu. Tyto metody pracují většinou na principu nahrazením určitého kódu jedním nebo skupinou symbolů a to tak, aby střední hodnota informace připadající na jeden symbol byla co

největší. Pokavaď se podíváme na nejběžnější kompresní formáty, tak i u toho nejlepšího se při snaze dosáhnout co největší komprese dat, se při určitém kompresním poměru začnou vždy objevovat nežádoucí obrazové artefakty. Je proto nutné nějakým způsobem ohodnotit, jak moc se míra rušení podepsala na komprimovaném obrázku. Hodnotit můžeme buďto subjektivní nebo objektivní metodou. Objektivní metody vycházejí ze vzorečků a jasně stanovených měření, ovšem pokavaď se nám výsledný obrázek nebude líbit a celkový dojem bude špatný, potom už na dobrých hodnotách objektivní metody nezáleží.

Základní a jednoduchá subjektivní metoda hodnocení rušení ve výsledném obrázku je metoda MOS (Mean Opinion Score). Z pětistupňové škály vybírá skupina dotázaných lidí tu hodnotu, která nejlépe popisuje rozdíl mezi původním a komprimovaným obrázkem. Jednotlivé výsledky se aritmeticky zprůměrují. Pětistupňová škála vyjadřující zkreslení je následující:

1	Silně rušící
2	Rušící
3	Lehce rušící
4	Postřehnutelný
5	Nepostřehnutelný

Obr. 5.1: Přehled rušení a jeho ohodnocení.

Výsledky této metody jsou sice přesné, ale velkou nevýhodou je její časová náročnost. Proto se dále budeme zabývat objektivními kriterii, které můžeme přenechat výpočetní technice.

5.2 OBJEKTIVNÍ KRITERIA

Existují dva základní typy. Jedním z nich je vyhodnocení rušivých artefaktů pouze z komprimovaného obrázku. Tyto metody jsou náročnější, ale v případě, kdy není k dispozici původní obraz, jsou nezbytné. Druhým typem jsou metody srovnávací. Tyto metody porovnávají rozdíly mezi obrazem původním a komprimovaným. Tyto metody se přímo nabízí pro testování kompresních algoritmů.

5.2.1 Střední absolutní chyba

Tento způsob měření je založený na rozdílu obrazových bodů na určité pozici mezi obrazem rekonstruovaným a původním. Tyto metody jsou sice jednoduché, ale o celkovém obrazu toho moc nevykazují, protože neberou v úvahu závislost mezi jednotlivými obrazovými body navzájem a tudíž se tak vzdalují lidskému vnímaní.

Střední absolutní chyba *MAE* (Mean Absolute Error) udává průměrnou hodnotu rozdílu mezi původním a komprimovaným obrázkem. Vztah pro výpočet MAE je:

$$MAE = \frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_k(m,n) - C_k(m,n)]$$

5.2.2 Střední kvadratická chyba

Střední kvadratická chyba *MSE* (Mean Square Error), je více používaným ukazatelem nežli MAE. Je definována podle vztahu:

$$MSE = \frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_k(m,n) - C_k(m,n)]^2$$

5.2.3 Odstup signál od šumu

Odstup signál od šumu *SNR* (Signal to Noise Ratio) znázorňuje podíl energie referenčního obrázku E_{ref} a MSE. Rovnice vypadá takto:

$$SNR = 10\log\frac{E_{ref}}{MSE} = 10\log\left(\frac{\frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_k(m,n)]^2}{\frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_k(m,n) - C_k(m,n)]^2}\right)[\text{dB}]$$

5.2.4 Špičkový odstup signálu od šumu

Ve zpracování obrazu je však nejčastěji udáván špičkový odstup signálu od šumu *PSNR* (Peak Signal to Noice Ratio), protože není závislý na energii původního obrázku, nýbrž pouze na střední kvadratické chybě. Místo energie původního obrázku

použijeme energii maximální E_{max}, tedy jako kdyby každý pixel měl hodnotu 255. Rovnice je definována jako:

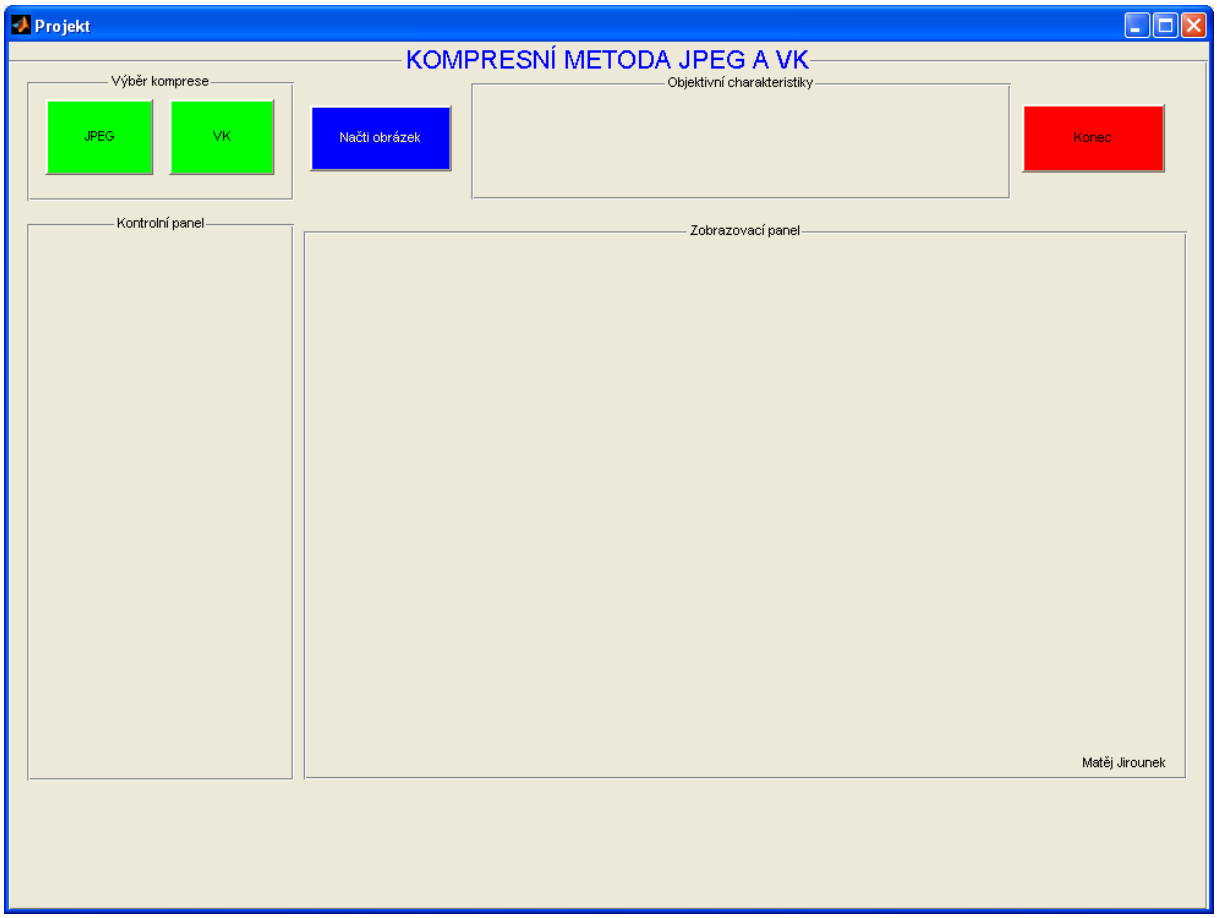
$$PSNR = 10\log\frac{E_{max}}{MSE}$$

$$= 10\log\left(\frac{\frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}255^2}{\frac{1}{MNK}\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N}[R_k(m,n)-C_k(m,n)]^2}\right)[\mathrm{dB}]$$

Tato metoda je dobrým měřítkem jak hodnotit jednotlivé kompresní metody, neda se na ni úplně spolehnout, protože né vždy naměřené hodnoty odpovídají výsledku. Toto je způsobené tím, že PSNR nebere v úvahu vazby mezi jednotlivými obrazivými body a protož ekaždá kompresní metoza zanáší do komprimovaného obrázku nějaké artefakty, takže nemůžeme uvažovat, že by rozložení chyby mezi obrazovými doby dvou obrázků bylo nezávislé.

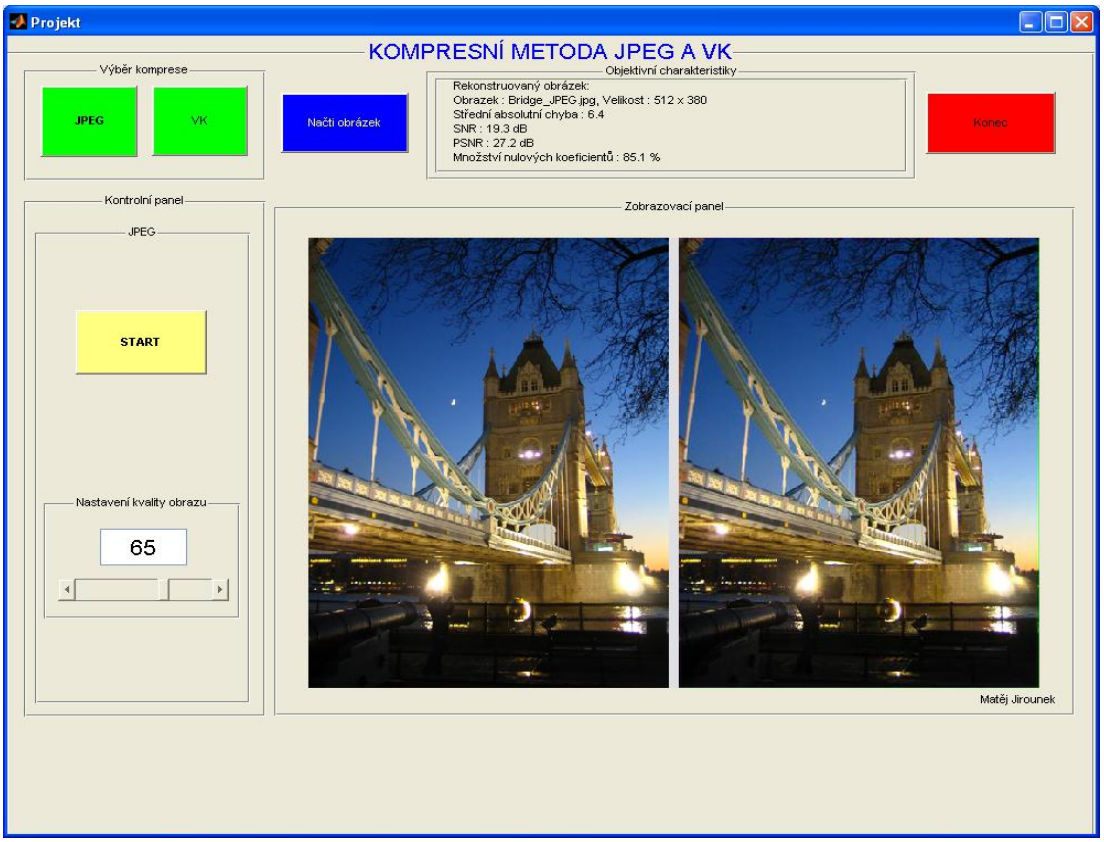
6 IMPLEMENTACE V PROGRAMU MATLAB

Po spuštění aplikace Projekt se nám objeví hlavní okno uživatelského rozhraní, které je pomocí panelů rozděleno na 4 základní části: Výběr komprese, Kontrolní panel, Zobrazovací panel a Objektivní charakteristiky.



Obr. 6.1: Úvodní okno.

Po načtení obrázku přes příslušné tlačítko „Načti obrázek“, se nám otevře dialogové okno, v němž vybereme požadovaný obrázek, se kterým chceme pracovat, a ten se zobrazí v zobrazovacím panelu. V panelu „Výběr komprese“ máme k dispozici dvě kompresní metody. Metodu JPEG a metodu Vektorová kvantizace (VK). Po výběru některé z těchto metod se v Kontrolním panelu zobrazí kontrolní části této metody. U metody JPEG je to slider pro nastavení požadované kvality komprese a tlačítko „START“. U druhé metody VK je to pak slider pro nastavení velikosti kódové knihy, panel pro výběr velikosti obrazového bloku (1x2, 2x2, 3x3) a samozřejmě tlačítko „START“.

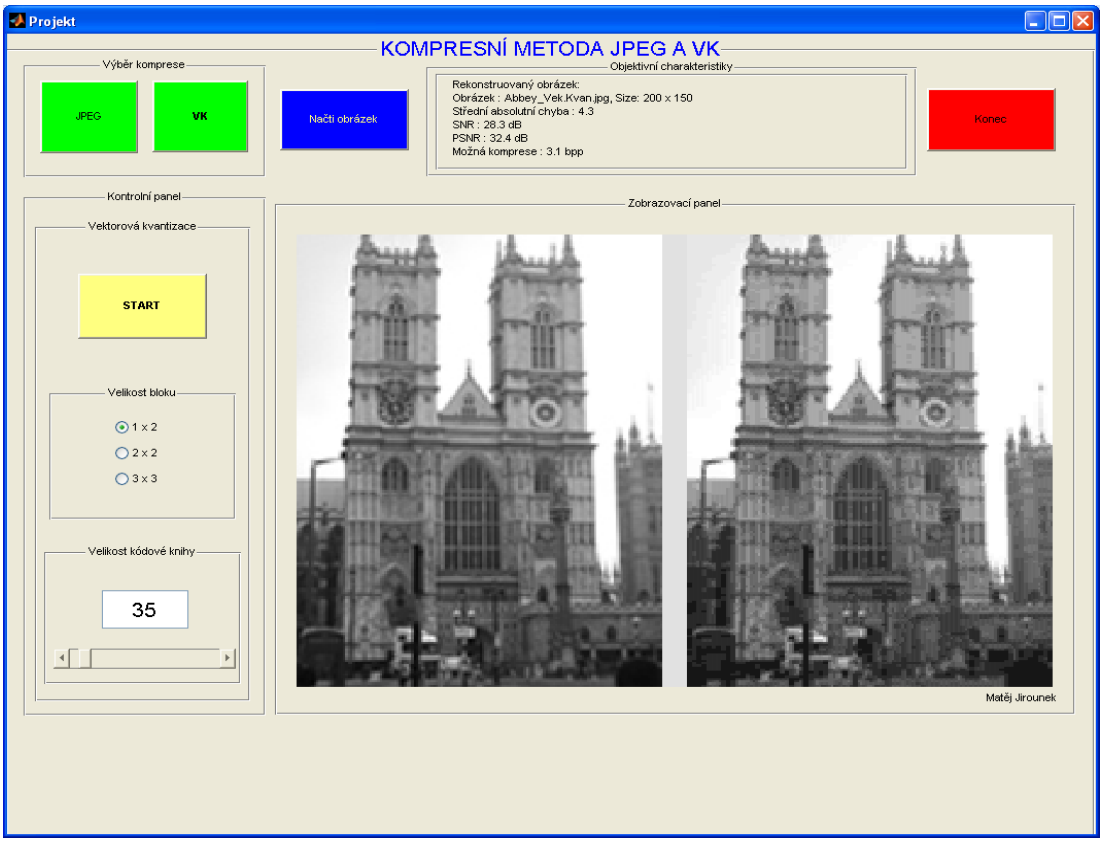


Obr. 6.2: JPEG komprese.

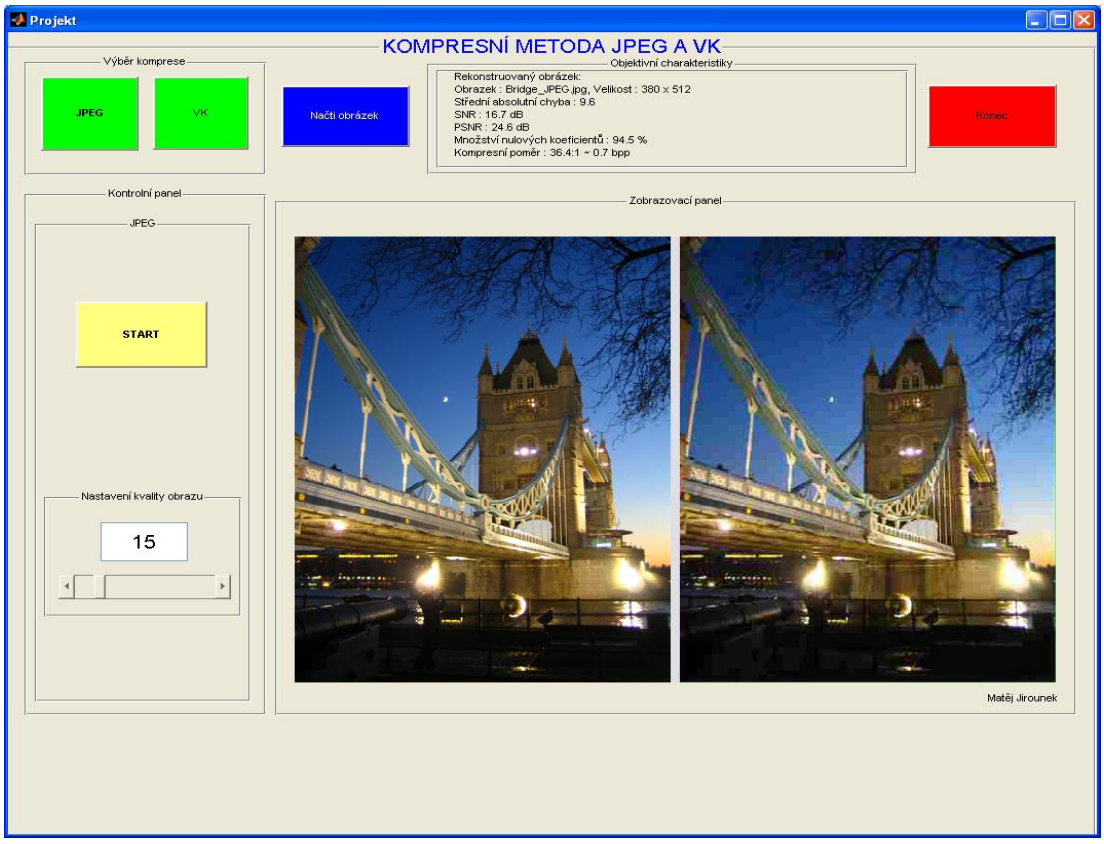
Po provedení nastavení a spuštění komprese se nám v Zobrazovacím panelu zobrazí jak obrázek původní, tak obrázek rekonstruovaný. Můžeme tedy snadno vlastním okem posoudit vliv komprese. Pro podrobnější porovnání obrázků nám slouží informace v panelu Objektivní charakteristiky. U komprese *JPEG* to jsou název a velikost obrázku a nadále to jsou charakteristiky *SNR* (Signal to Noise Ratio), tedy odstup signálu od šumu a *PSNR* (Peak Signal to Noise Ratio), v překladu špičkový odstup signálu od šumu. Prve zmiňovaná charakteristika *SNR* udává podíl energie referenčního obrázku E_{ref} ke střední kvadratické chybě *MSE* (Mean Square Error). *SNR* charakteristika se udává v decibelech [*dB*].

Ve zpracování obrazu se však častěji používá charakteristika *PSNR*. Není totiž závislá na energii referenčního obrázku, ale pouze na střední kvadratické chybě. Na místo energie referenčního obrázku je zde počítáno s energii maximální E_{max}, tedy jako kdyby každý koeficient pixelu měl hodnotu 255.

U metody VK je to obdobné s tím rozdílem, že místo údaje o množství nulových koeficientů se zobrazí údaj o možné kompresi.



Obr. 6.3: VK komprese.



Obr. 6.4: Po dokončení komprese

6.1 UKÁZKY NĚKTERÝCH ČÁSTÍ KÓDU

- Zobrazení polí JPEG a VK.

```
function pbJPG_Callback(hObject, eventdata, handles)
set(handles.panelJPG,'Visible','on');
set(handles.PanelObjJPEG,'Visible','on');
set(handles.panelVQ,'Visible','off');
set(handles.PanelObjVK,'Visible','off');
set(handles.pbJPG,'FontWeight','bold');
set(handles.PanelObjJPEG,'FontWeight','bold');
set(handles.pbVQ,'FontWeight','normal');
set(handles.PanelObjVK,'FontWeight','normal');

function pbVQ_Callback(hObject, eventdata, handles)
set(handles.panelVQ,'Visible','on');
set(handles.PanelObjVK,'Visible','on');
set(handles.panelJPG,'Visible','off');
set(handles.PanelObjJPEG,'Visible','off');
set(handles.pbVQ,'FontWeight','bold');
set(handles.PanelObjVK,'FontWeight','bold');
set(handles.pbJPG,'FontWeight','normal');
set(handles.PanelObjJPEG,'FontWeight','normal');
```

- Blok VK a výběr bloku.

```
function rbVQ12_Callback(hObject,eventdata,handles)
set(handles.sliderVQ,'Value',4);
set(handles.editVQ,'String',4);
fullname=get(findobj('Tag','figure1'),'UserData');
[slidermax]=f_sliderVQmax(fullname,1,2);
set(handles.sliderVQ,'Max',slidermax);
function rbVQ22_Callback(hObject,eventdata,handles)
set(handles.sliderVQ,'Value',4);
set(handles.editVQ,'String',4);
fullname=get(findobj('Tag','figure1'),'UserData');
[slidermax]=f_sliderVQmax(fullname,2,2);
set(handles.sliderVQ,'Max',slidermax);
function rbVQ33_Callback(hObject,eventdata,handles)
set(handles.sliderVQ,'Value',4);
set(handles.editVQ,'String',4);
fullname=get(findobj('Tag','figure1'),'UserData');
[slidermax]=f_sliderVQmax(fullname,3,3);
set(handles.sliderVQ,'Max',slidermax);
```

- Zobrazení charakteristik JPEG.

```
    set(handles.textJPEGobj,'String',{
    ['Rekonstruovaný obrázek:'];...
    ['Obrazek : ',filename,', Velikost : ',num2str(size(IMG,2)),' x
',num2str(size(IMG,1))];...
    ['Střední absolutní chyba : ',num2str(round(10*r_prumer)/10)];...
    ['SNR : ',num2str(round(10*SNR)/10),' dB'];...
    ['PSNR : ',num2str(round(10*PSNR)/10),' dB'];...
    ['Množství nulových koeficientů : ',num2str(pocetNul),' %'];...
    ['Kompresní poměr : ',num2str(komprese),':1 ~
',num2str(round(10*(24/komprese))/10), ' bpp']});
```

- Výběr kvality komprese JPEG.

```
if (kvalita==50)
        Q=Q50;
    elseif ((kvalita>=1)&&(kvalita<50))
        Q=Q50*50/kvalita;
    elseif ((kvalita>50)&&(kvalita<=100))
        Q=Q50*(100-kvalita)/50;
end;
```

- Přepočet barevného modelu z RGB na YCbCr.

```
% Prepocet RGB na YCbCr
Y=0.299*R+0.587*G+0.114*B;Y=round(Y);
Cb=-0.169*R-0.331*G+0.5*B+128;Cb=round(Cb);
Cr=0.5*R-0.419*G-0.081*B+128;Cr=round(Cr);
```

- Kvantizační matice Q50 černobílý.

```
Q50_8x8=[16  11  10  16  24  40  51  61
         12  12  14  19  26  58  60  55
         14  13  16  24  40  57  69  56
         14  17  22  29  51  87  80  62
         18  22  37  56  68 109 103  77
         24  35  55  64  81 104 113  92
         49  64  78  87 103 121 120 101
         72  92  95  98 112 100 103  99
        ];

Q50=Q50_8x8;
```

- Kvantizační matice Q50 barevný.

```
Q50CbCr_8x8=[17  18  24  47  99  99  99  99
             18  21  26  66  99  99  99  99
             24  26  56  99  99  99  99  99
             47  66  99  99  99  99  99  99
             99  99  99  99  99  99  99  99
             99  99  99  99  99  99  99  99
             99  99  99  99  99  99  99  99
             99  99  99  99  99  99  99  99
            ];

Q50CbCr=Q50CbCr_8x8;
```

Převod na dvouprvkové vektory.

```
dvojice=zeros(ceil(r/2)*s,2);
for ii=1:r
    for jj=1:s
         if (mod(jj,2)~=0)       %lichy sloupec
             dvojice((ii-1)*ceil(s/2)+ceil(jj/2),1)=IMG(ii,jj);
         else                    %sudy sloupec
             dvojice((ii-1)*ceil(s/2)+ceil(jj/2),2)=IMG(ii,jj);
         end;
    end;
end;
```

7 ZÁVĚR

V této práci jsem měl za úkol shrnout v současnosti používané kompresní techniky, především JPEG a JPEG2000 a dále v prostředí MATLAB implementovat tyto kompresní metody. Výstupem této práce je aplikace v které jsem implementoval formát JPEG, konkrétně ztrátový algoritmus standardu JPEG a základní typ vektorová kvantizace s adaptivní kódovou knihou. Po nastavení jednotlivých parametrů program provede odpovídající kompresi a zobrazí základní objektivní měřítka – střední absolutní chybu, SNR, PSNR, množství nulových koeficientů a kompresní poměr. Tyto objektivní charakteristiky jsem také popsal v kapitole 5.2. Dále jsem se ve třetí kapitole zabýval bezztrátovými metodami komprese, zejména však Huffmanovým kódováním a algoritmy LZW. Ve čtvrté kapitole jsou rozebrany ztrátové kompresní metody, tedy diskrétní kosínova transformace, diskrétní vlnková transformace a vektorová kvantizace, zejména však standardy JPEG a JPEG2000. Chtěl bych podotknout, že srovnání rekonstruovaných obrázků v kapitole 4.5 jsem provedl prostřednictvím programu Adobe Photoshop. Díky výstupům z aplikace a popisu charakteristik si můžeme snadno udělat srovnání těchto kompresních metod a to jak objektivními, tak i subjektivními kritérii.

Závěrem bych chtěl ještě nastínit, jakým směrem by se mohla dale rozvíjet tato práce. Zajímavé by bylo nastudování format JPEG2000, jedo implementace v prostředí MATLAB a jistě i příme srovnaní s formátem JPEG v této aplikaci.

Literatura

[1] MALÝ J.: Metoda pro kompresi obrazových signálu pomocí waveletové transformace, Diplomová práce, Ústav Telekomunikací FEKT VUT Brno, 2006.

[2] Matlab *HELP* [Nápověda k programovému prostředí Matlab®]. Natick (Massachusetts, USA): The MathWorks Inc.

[3] Wikipedia, The Free Encyclopedia, *Discrete cosine transform*, 8. května 2007. Dokument dostupný na URL.

[4] Wikipedia, The Free Encyclopedia, *Discrete wavelet transform*, 8. května 2007. Dokument dostupný na URL.

[5] Bodeček K.; *Škálovatelná komprese videa*; Elektrorevue - Internetový časopis (http://www.elektrorevue.cz), vol. 2006

[6] ZAPLATÍLEK K., DOŇAR B. - MATLAB / Tvorba uživatelských aplikací; BEN – technická literatura, Praha 2004

[7] Shi, Y. Q. Image and video compression for multimedia engineering / New York : CRC Press, 2000. 480 s. ISBN 0-8493-3491-8

[8] MARCHAND P., HOLLAND O.T. – Graphics and GUIs with MATLAB, CRC Press 2003

Seznam použitých veličin, symbolů a zkratek

Veličiny a symboly:

P(x)	[-]	pravděpodobnost
s(n)	[-]	proměnná, posloupnost
m(n)	[-]	proměnná, posloupnost
X_i	[-]	proměnná, událost
m	[-]	proměnná, řetězec
i	[-]	index obrazového bodu
j	[-]	index obrazového bodu
u	[-]	index spektrálního koeficientu
v	[-]	index spektrálního koeficientu
f(m, n)	[-]	amplitůda prvku obrázku
M	[pixel]	rozměr obrázku
N	[pixel]	rozměr obrázku
C_r	[-]	kompresní poměr
Q_{50}	[-]	kvantizační matice
K	[-]	počet obrazových složek
R_k	[-]	bod k-té složky referenčního obr.
C_k	[-]	bod k-té složky komprimovaného obr.
MAE	[-]	střední absolutní chyba
MSE	[-]	Střední kvadratická chyba
SNR	[dB]	Odstup signal od šumu
PSNR	[dB]	Špičkový odstup signal od šumu
E_{ref}	[-]	Energie referenčního obrázku
E_{max}	[-]	Maximální energie referenčního obrázku

Zkratky:

JFIF	***J**PEG **F**ile **I**nterchange **F**ormat*
JPEG	***J**oint **P**hotographic **E**xperts **G**roup*
JPEG2000	***J**oint **P**hotographic **E**xperts **G**roup 2000*
RLE	***R**un-**L**ength **E**ncoding*
LZ	***L**empel-**Z**iv*
LZ77	Následující verze LZ
LZ78	Následující verze LZ78
LZW	***L**empel-**Z**iv-**W**elch*
ARJ	***A**rchived by **R**obert **J**ung*
RAR	***R**oshal **AR**chive*
GIF	***G**raphics **I**nterchange **F**ormat*
TIFF	***T**ag **I**mage **F**ile **F**ormat*
PNG	***P**ortable **N**etwork **G**raphics*
RGB	***R**ed, **G**reen, **B**lue*
$Y'C_bC_r$	Y – luminale (jas), Cb – modrý chrominanční komponent, Cr – červený chrominanční komponent
DFT	***D**iscrete **F**ourier **T**ransform*
DCT	***D**iscrete **C**osine **T**ransform*
DCT2	Dvojrozměrná ***D**iscrete **C**osine **T**ransform*
STFT	***S**hort-**T**ime **F**ourier **T**ransform*
HP	Horní propust
DP	Dolní propust
VK	Vektorová kvantizace
RCT	***R**eversible **C**omponent **T**ransformation*
EBCOT	***E**mbledded **B**itplane **C**oding with **O**ptimal **T**runcation*
MOS	***M**ean **O**pinion **S**core*
MAE	***M**ean **A**bsolute **E**rror*
MSE	***M**ean **S**quare **E**rror*
SNR	***S**ignal to **N**oise **R**atio*
PSNR	***P**eak **S**ignal to **N**oise **R**atio*
LRU	***L**ast **R**ecently **U**sed*

Seznam příloh

1. **DVD – Obsah:**

- elektronická verze bakalářské práce (ve formátu pdf)
- složka Program – Vlastní aplikaci Projekt
  Soubory potřebné k aplikaci Projekt
  Složku Obrázky – testovací obrázky